Методичка 8.2 - Знакомство с дизассемблером IDA. Часть 1

Зачем нужен дизассемблер?

На данный момент дизассемблер IDA Pro является, наиболее мощным дизассемблером, представленном на рынке. Под мощностью я подразумеваю уровень анализа дизассемблированного кода, удобство работы с кодом, количество различных плагинов, которые ускоряют процесс анализ кода, а также уровень декомпилятора. В совокупности перечисленных характеристик, по моему мнению, дизассемблер IDA Pro сейчас находится впереди конкурентов.

Если вы планируете развиваться в области реверс-инжиниринга и стать, например, вирусным аналитиком, то навыки работы с IDA Pro вам однозначно будут полезны.

IDA Pro - это интерактивный дизассемблер и отладчик одновременно. Он позволяет превратить бинарный код программы в ассемблерный текст, который может быть применен для анализа работы программы.

Название IDA Pro происходит от английского Interactive Disassembler. Дизассемблер IDA Pro используют в основном антивирусные компании для анализа различного вредоносного программного обеспечения, а также пентестеры для исследования защиты различных систем. IDA Pro также может поставляться с декомпилятором, который называется HexRays. Декомпилятор HexRays позволяет выполнять трансляцию исполняемого бинарного файла в эквивалентный исходный код на языке Си. Использование декомпилятора сильно ускоряет процесс анализа и экономит время, так как анализировать код на высокоуровневом языке сильно проще, чем на языке Ассемблер.

Дизассемблер IDA Pro в основном используется для анализа вредоносного ПО, поиска уязвимостей и для взлома программ.

Виды лицензий IDA

- *Generalities*
  - *Бессрочная лицензия*
- *Floating Licenses*
  - *Возможна установка на несколько компьютеров*
- *Computer Licenses*
  - *Возможна установка только на один компьютер*
- *Named Licenses*
  - *Именная лицензия*
- *Educational Licenses*
  - *Университетская лицензия. Бесплатная, но без технической поддержки*

Бесплатная версия IDA Freeware
https://www.hex-rays.com/products/ida/support/download_freeware.shtml

Как вы возможно знаете, дизассемблер IDA Pro является платным программным обеспечением. На сайте производителя можно ознакомиться с различными видами лицензий. Краткое их описание вы можете увидеть на слайде. В рамках курса мы будем использовать бесплатную версию дизассемблера IDA Freeware.

Рекомендации по установке дизассемблера IDA Freeware

Ссылка: https://www.hex-rays.com/products/ida/support/download_freeware.shtml

Обзор базовых функций IDA Freeware

Бесплатная версия IDA Freeware позволяет анализировать бинарные файлы с архитектурой x64. В прошлом видео мы уже создали подобный бинарный файл и сейчас попробуем его проанализировать средствами дизассемблера IDA Freeware.

Запускаем с рабочего стола IDA Freeware и выбираем файл, который был создан в предыдущем видео.

New - prog-6.1.exe

Мы видим, что IDA сама определила, что выбранный нами файл имеет PE-заголовок и архитектуру x64. Так как IDA получила всю информацию о бинарном файле из PE-заголовка, нам не нужно ничего больше заполнять, нажимаем OK.

На данном этапе IDA уже выполнила автоматический анализ файла. Слева мы видим список функций, которые имеются в файле. В этом списке присутствуют не только наши функции, но и системные функции. Так как наша программа была скомпилирована без отладочных символом, то IDA не может знать как называются функции, поэтому она даем им название исходя из их адреса (например, sub_1400014C8). Если мы посмотрим на список функций более внимательно, то увидим функцию с именем start. Эту функцию так назвала IDA, так как эта функция находится по адресу, куда указывает точка входа из PE-заголовка. Это значит, что программа начинает выполняться именно с этой функции.

Давайте выберем эту функцию в списке функций и в главном окне увидим набор инструкций, который в ней имеется. Инструкций совсем не много. Сначала мы увеличиваем стек и вызываем какую-то функцию, затем уменьшаем стек и выполняем прыжок по какому-то адресу.

Если мы дважды кликнем по имени первой функции, то автоматически перейдем на эту функцию.

Здесь мы видим вызовы каких-то системных функций:

GetCurrentThreadId
GetCurrentProcessId

QueryPerformanceCounter

Вызова других функций мы здесь не видим, а нам неплохо было бы найти нашу функцию main() среди нескольких сотен функций.

Чтобы вернуться назад, достаточно нажать на кнопку “Esc” и мы вернемся на функцию, на которой были до перехода. Итак, мы снова в функции start.

В конце функции мы видим инструкцию jmp. Давайте перейдем по указанному адресу и узнаем, что там находится.

Мы видим, что здесь кода гораздо больше, чем в предыдущей функции. Чтобы понять, насколько много блоков в функции и уменьшить масштаб, можно нажать кнопку “W”, что означает Fit window. Для того, чтобы вернуться к 100% масштабу нужно нажать на кнопку - 1.

Мы видим, что в этой функции очень много переходов по другим адресам и перебирать их все в поисках функции main является не самым эффективным способом.

Когда мы запускаем нашу программу в командной строке:

>prog-6.1.exe

Usage: prog-6.1.exe <license key>

> prog-6.1.exe AAAA

ERROR. License key should contains 12 symbols.

> prog-6.1.exe AAAABBBBCCCC

ERROR. License key is incorrect.

Мы видим некоторые строки. IDA в процессе анализа бинарного файла также выполняет поиск строк в файле и может их нам показать в специальной вкладке. Давайте добавим вкладку Strings.

Все вкладки расположены над главным экраном: IDA View-A, Hex View-1 и т.д. Добавить вкладку со строками можно через меню.

View - Open subviews - Strings (или Shift + F12)

Теперь мы видим все строки, которые смогла найти IDA. Среди них есть и строки, которая печатала программа в консоли, например, “ERROR. License key is incorrect.”. В колонке Address мы видим адрес этой строки и указание на то, что строка находится в секции данных

(.rdata:000000014001A300). Если строк в списке очень много и, просто пролистывая, сложно найти нужную строку, то можно набрать первые буквы и IDA сама найдет ей (набираем \n ERROR).

Выбираем строку и нажимаем Enter и нас перекидывает на вкладку IDA-View-A. Сейчас, судя по адресу, мы находимся в секции данных. Рядом с нашей строкой расположены и другие строки. Для того, чтобы узнать, в какой функции используется наша строка мы можем спросить об этом у IDA. Если мы кликнем на имя строки aErrorLicenseKe и нажмем клавишу - “X”, то IDA покажем нам все места в коде, где есть обращение к этой строке. Это называется Xref, что переводится, как перекрестные ссылки.

Выбираем функцию, где используется наша строка и нажимаем OK. Далее нас перекидываем в функцию, где мы видим и другие строки нашей программы, например, "ERROR. Your license key is invalid.\n". Делаем вывод, что мы попали в функцию main.

Если мы уверены, что это функция main, то мы можем ее переименовать и она переименуется во всем проекте. Для этого поднимаемся в начало функции и ищем строку - “sub_140001000 proc near”. Нажимаем на текущее название функции и нажимаем клавишу - N и указываем имя - main.

В общем списке функций, также появилась функция с именем main.

Как вы видите, дизассемблер за нас определил порядок передачи аргументов в программе - это cdecl. Также он добавил названия аргументов argc и argv.

В следующем видео этого урока мы продолжим реверс-инжиниринг программы prog-6.1.exe.